PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G10H 7/02

A1

(11) International Publication Number: WO 99/16049

(43) International Publication Date: 1 April 1999 (01.04.99)

(21) International Application Number: PCT/SG98/00074

(22) International Filing Date: 22 September 1998 (22.09.98)

(30) Priority Data:
9703500-0 23 September 1997 (23.09.97) SG

(71) Applicant *(for all designated States except US)*: KENT RIDGE DIGITAL LABS (KRDL), NATIONAL UNIVERSITY OF SINGAPORE [SG/SG]; Heng Mui Keng Terrace, Kent Ridge, Singapore 119597 (SG).

(72) Inventors; and

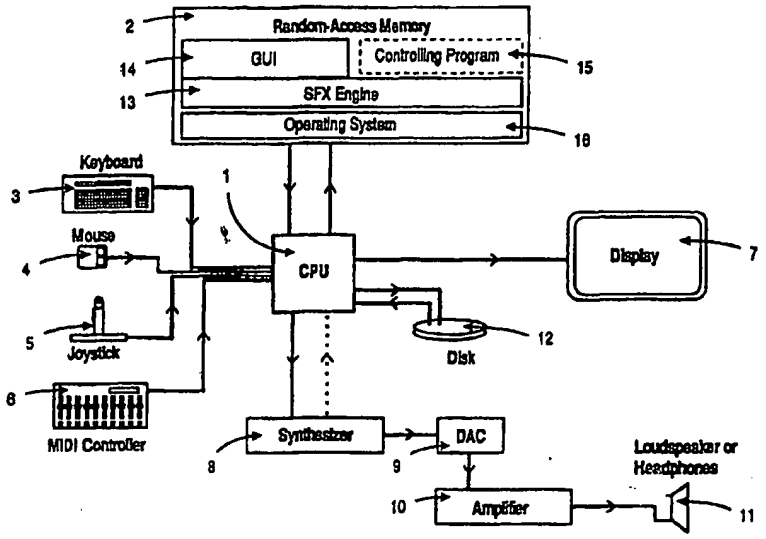
(75) Inventors/Applicants *(for US only)*: WYSE, Lonce LaMar [US/SG]; Block 3, Normanton Park, #09-175, Singapore 119000 (SG). KELLOCK, Peter, Rowan [GB/SG]; 113 Clementi Road #09-06, Block D Kent Vale, Singapore 129793 (SG).

(74) Agent: TEAN, Ng, Kim; Legal Counsel (Intellectual Property), Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613 (SG).

(81) Designated States: AU, JP, US, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: INTERACTIVE SOUND EFFECTS SYSTEM AND METHOD OF PRODUCING MODEL-BASED SOUND EFFECTS



(57) Abstract

The present sound effects system produces sound-effects which are based on parameterized models of sound-generating phenomena such as footsteps, earthquake, etc. The behavioral characteristics of the phenomena are taken into consideration when creating the models to ensure realism. Because the sound effects are created from models, the system allows a user greater flexibility and interactivity. By controlling a set of parameters, the user can interact with the system in real time. The system is versatile in that it can produce a wide range of sound-effects, and is not limited to producing any particular sound effects or a class of sound effects.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Türkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

INTERACTIVE SOUND EFFECTS SYSTEM AND METHOD OF PRODUCING MODEL-BASED SOUND EFFECTS

FIELD OF THE INVENTION

The present invention relates generally to the field of interactive media and sound production. In particular, it relates to the modeling and production of interactive model-based sound effects, and a system which can produce wide range of sound effects.

BACKGROUND OF THE INVENTION

Mechanical inventions for creating sound effects have been used for centuries, for example, the metal thunder-sheet, and barrel+rope "lion's roar" used in theater. In the twentieth century, sound effects specialists such as Russolo (The Art of Noises, 1916) pioneered the use of electromechanical devices for producing noises in musical or quasi-musical performances. The pioneers of film created a plethora of techniques and tricks for creating sound effects, including the "Foley" technique of performing sound effects by manipulation of physical objects in synch with the events on a film.

Since the arrival of digital audio, sound effects specialists have used computers to edit and process recorded sound effects using tools such as digital audio editors. Hardware and software designed primarily for musical purposes (hardware such as MIDI keyboards & slider banks, software such as sequencers) are also widely used in the production of sound effects.

Sound effects are widely used in the traditional media such as film, television, radio and theater, as well as in the interactive media such as computer games, multimedia titles, and World-Wide-Web sites. And while the effects are often produced in similar ways, how they are used may differ greatly depending on the medium. This is because the sound effects remain in a fixed form in the traditional media while the interactive media require the sound effects to vary according to a given situation. A sound effect in a film, for instance, remains the same from one viewing to the next; there is no interaction between the sound effects and the viewer. In the case of the interactive media such as computer games, there are multiple possibilities or "paths" through the production depending on the user's actions at each moment in time. Hence, both the visual and audible elements must constructed from an underlying model at the moment they are called upon if the user's experience is to be highly interactive.

In most computer games and virtual environments today, the visual elements are largely constructed from models, thus providing a high degree of interactivity in the visual experience. A model-based image is different from a prerecorded image in that the former is constructed from a set of parameters which are adjustable; the later is fixed and cannot be easily manipulated. For example, objects move in the virtual space when commanded to do so by the user, and the angle of view of a scene will change as the user turns in the virtual space similar to how a person's view changes as he turns his head in the real world. This is possible even though not all views have been pre-recorded.

In the case of sound effects, however, today's solution is generally to retrieve pre-recorded audio segments which typically last from one second

to several minutes. For instance, consider a video game which simulates a car in motion. Whenever the car experiences an event which would produce a sound in real life, e.g., hitting an obstacle at a high speed, a pre-recorded sound is played at the precise moment the collision occurs. Typically, the relationship between the sound and the event, i.e., collision, is very crude; that is, the sound effect does not take into account the speed of the car, the material of the object it collides with, the angle of impact, or the reverberation characteristics of the surroundings, etc., for instance. While it is possible to take into account different factors by playing different pre-recorded sound segments corresponding to the particular situation at hand, it is impractical or often impossible to collect a complete continuum of sounds that would precisely correspond to every possible scenario.

Producing sound effects by recalling pre-recorded sound segments has the limitation that the existing sound segments cannot easily be combined or manipulated in a seamless manner to produce a variation of the original sound segments which closely simulates the subtleties of the real-world conditions. This is because pre-recorded sounds are not based on a model consisting of parameters of the sound-producing phenomenon. So if one were to want to produce a sound effect which accurately models a real-life event for which a sound segment is not available, a totally new segment of sound must be cumbersomely recorded; it cannot be produced by combining or manipulating the existing segments.

Pre-recording of sounds is how virtually all sound effects are currently produced in both the traditional and the interactive media. In the interactive media, the main shortcoming of this method is the lack of realism as discussed above. For the traditional media, the shortcoming is increased

labor. For example, if a series of footsteps is needed in a film, a sound effect specialist would have to view a particular scene in the film then create and record a series of actual footstep sounds (however he chooses to do so whether by actually walking or banging physical objects together) that exactly correspond to what appears in the scene. If for some reason, the scene is changed from a walking scene to a running scene, a new recording session would have to be conducted in order to produce footsteps which correspond to the increased speed of the steps. It wouldn't be possible to produce realistic-sounding footsteps merely by manipulating the previously recorded footsteps from the walking scene.

One solution to this problem in the traditional media was to have a collection of many samples, usually provided on CDs or CD-ROMs. Although these collections can include thousands of samples, this solution has still posed many problems. For one, it is inflexible: the samples cannot be easily manipulated. In addition, it is often time consuming to find the right sample, since a person needs to go through many samples and check for acceptability. But the most critical problem is that despite the high number, it is still difficult the find the precise sample which fits the purpose for which is sound effect is being used given the infinity number of variations in the sound effects needed. Hence, it can be seen that there is a great need in the sound effect industry to have a model-based method and system for producing sound effects which can both capture realism as well as save time and labor.

Although there exists several sythesis techniques, none meet the needs described above. There are currently several distinct clusters of technology for digital sound synthesis. One is dedicated synthesizers with

specialized circuitry and/or DSP code which are driven by hardware and/or software controllers for real-time generation and interaction. A second cluster is general purpose software synthesizers capable of implementing any conceivable synthesis algorithm. They are generally not real-time, but have some kind of compiling stage which produces a sound file.

Computer sound as experienced by most people is produced using a soundcard on a desktop computer. Typically a small number of synthesis methods are available and control is via the MIDI protocol, with the host CPU, another networked computer, or an external device such as a musical keyboard, functioning as the controller.

There are three synthesis methods commonly used on today's commercial soundcards, frequency modulation (FM), wavetable, and waveguide (a kind of "physical modeling"). Whereas wavetable synthesis typically relies on recorded material (e.g. notes of an instrument), FM and waveguide synthesis are purely synthetic. Purely synthetic algorithms provide more flexibility in the sounds they produce, but the imitations of natural sounds tend not to be as realistic as wavetable methods based on recordings of the modeled sound source. Wavetable synthesis usually involves some manipulation of the recorded sounds during execution such as looping sections of the sound, pitch shifting, and adjusting onset and decay times. These standard techniques provide the flexibility commonly used in modeling a wide range of a musical instrument's capabilities based on only a small number of different recorded notes. Simple unprocessed playback of prerecorded clips is only trivially "synthesis" but it is currently the method most commonly used for generating sound effects within software environments

One can make a distinction between several different time scales for sound synthesis. One is the rate at which samples must be delivered to the audio output device(e.g. 44.1 KHz for a channel of "CD quality" sound). Another is the control rate at which parameter values controlling the synthesis must be updated. This can often be slower than the sample rate. For example, at the rate that pitch and volume changes usually occur in instrumental and natural sounds, the synthesis routine can usually be updated with these parameters an order of magnitude or two slower than the sample rate and still create the perception of smooth variation. Finally, one can consider an event time scale based on the rate at which synthesis algorithms (e.g. musical notes) are started and stopped.

The distinction between the different time scales is somewhat arbitrary, and even inappropriate for certain synthesis paradigms, but it is useful for many. The MIDI protocol is based on the distinction between controller and synthesizer and has made the most of a common bus architecture with a transfer rate of about 3K bytes/second. In the context of the networked computers in which shared virtual environments and multi-user games are commonplace, communication bandwidth is similarly at a premium, while client computers are powerful and quite capable of handling a significant amount of synthesis. If clients share a common library of synthesis routines, then an "event and control parameter" representation of a sound is extremely efficient, puts a low demand on communications bandwidth, and is capable of generating arbitrarily high sound quality depending upon the capabilities of the client.

The considerations outlined above have led to present sound effects modeling based on wavetable event patterns. The fact that individual

events are still based on recorded sound samples allows the present system to retain a significant degree of the realism associated with the playback of recorded audio and wavetable synthesis, but the move toward synthetic construction in the pattern representation allows the system to move up the curve in flexibility.

Since a common desktop configuration includes a soundcard with wavetable capabilities, so that many of the sound processing and mixing operations are performed on the card, the host is burdened with sound construction only at the event control rate. Even in a purely software synthesis environment, the wavetable event-pattern technique is more flexible and memory-efficient than audio playback, and more computationally efficient than sample-by-sample synthesis. This is significant when sounds are embedded in CPU-intensive applications.

Besides efficiency and expressivity, the other advantage of a modeling approach is the breadth of sounds that can be effectively represented within its confines. There are no sounds that fall out of the domain of sound effects, so generality is important.

The wavetable event-pattern representation falls between two extremes. On one end is a single long event, which obviously does not demonstrate the effectiveness of the representation. On the other extreme are large numbers of very short events. Granular synthesis is a general technique capable of representing any time domain signal, but this end of the spectrum does not achieve the desired reduction in computation, bandwidth and memory that is desirable. It is preferred that the rate of events and synthesizer commands be kept low (the sounds rarely use more than 25 per second) to keep computation and bandwidth requirements

manageable, especially when several sounds processes are running at once. On the other hand, the shorter the atomic event elements are, the more flexibility we can provide in controlling the sounds. Quite effective interactivity can often be achieved with event rates on the order of 10's per second per sound effect.

Consider footsteps as an example. The desired control may involve the rate of walking or running, the weight of the person, the type of shoe (if any) being worn and surface characteristics. The event rate is manifestly low (though we actually use several events to represent a single step). The number of wavetables is some function of the desired variability, but a considerable amount of the variability can also be achieved with combinations of standard wavetable parameters such as volume and pitch.

Other event-oriented sound effects which are naturally modeled this way are applause, machine guns, and bouncing objects. Perhaps less obvious are sounds made of faster discrete events. This category includes card shuffling, certain bird chirping patterns, and some types of machinery such as ratchets, jackhammers and engines.

Another class of sound types that can be modeled with these techniques is textured sounds. In this category are waterfalls, wind, room tone, crowds, traffic, scraping, and rolling sounds. In most multimedia applications today, generating statistically stable sounds is done by creating loop points in a short recording. Unfortunately, the ear is very sensitive to the presence of such loops, the detection of which quickly destroys any sense of realism. Statistical event generating algorithms can be effectively used in most situations where loops are currently used; they

also keep the amount of stored data to a minimum, but do not produce the distracting looping effect.

Sounds that are perceived as single events (e.g. a gun shot or a pencil breaking) can also be modeled using synchronous or overlapping multiple wavetable events with flexibility being the prime advantage over simple audio playback, (memory usage could even be greater than a single event recording). Sounds that are more difficult, but not impossible, to model are those that do not have enough texture variation to permit undetected event changes (e.g. the whir of a drill). Sounds that are impractical to model this way are those that have complex characteristic temporal sequences across many attributes (e.g. speech).

The range of sounds for which this wavetable event-pattern representation is effective in terms of flexibility, memory usage and sound quality, is thus very broad.

OBJECTS OF THE INVENTION

It is therefore an object of the present invention to provide a sound effect system which responds to a user's actions in real time.

It is another object of the present invention to provide a sound effects system which can produce realistic-sounding sound effects.

It is yet another object of the present invention to provide a sound effects system which can substantially save time and labor in creating sound effects.

It is still yet another object of the present invention to provide a method and system for producing model-based sound effects.

It is still another object of the present invention to provide a method and system which is not constrained to producing a limited class or sub-class of sound effects, but which can produce a wide range of sound effects.

SUMMARY OF THE INVENTION

The present sound effects system produces sound-effects which are based on parameterized models of sound-generating phenomena such as footsteps, earthquake, etc. The behavioral characteristics of the phenomena are taken into consideration when creating the models to ensure realism. Because the sound effects are created from models, the system allows a user greater flexibility and interactivity. By controlling a set of parameters, the user can interact with the system in real time. The system is versatile in that it can produce a wide range of sound-effects, and is not limited to producing any particular sound effects or a class of sound effects.

The system includes a central processing unit is connected to random-access memory (RAM), one or more input devices such as keyboard, mouse, joystick, MIDI controller; a visual display device; a sound synthesizer; and audio output system including digital-to-analog converter (DAC), amplifier, loudspeaker or headphone (or alternatively, a soundcard integrating some of these individual devices can be used); and a non-

volatile storage device such a hard disk. These components play a supporting role to the central element of the invention, the interactive sound effects computer program (SFX program) which consists of a sound effects software engine (SFX engine) and optionally a graphical user interface program (GUI). In one mode of operation the GUI can be replaced by a controlling program which replaces the GUI. Also present is an operating system program, such as the standard operating system of any personal computer system.

The CPU executes the stored instructions of the programs in memory, sharing its processing power between the SFX engine, the GUI or controlling program, the operating system and possibly other programs according to a multitasking scheme such as the well-known time-slicing technique. Under command of the SFX engine, it delivers a stream of events or commands to the sound synthesizer, which produces digital audio in response to these commands. The output of the synthesizer is a digital audio signal which is converted to an analogue form by the digital to analogue converter (DAC), then amplified and delivered to the user by means of the amplifier and loudspeaker or headphones. Optionally the digital audio signal may also be delivered back to the CPU allowing it to be further processed or stored for later retrieval.

The SFX engine is controlled directly by means of the GUI, or from an external controlling program such as a computer game or rarely by both at the same time. When under control of the GUI, the user effectively interacts directly with the SFX program, controlling it by means of one or more input devices such as an alphanumeric computer keyboard, a pointing device, a joystick, a specialized controller such as a slider bank or music keyboard

connected by means such as the Musical Instrument Digital Interface (MIDI) standard, or other physical controlling means. In this mode of use, the GUI program uses the display device to provide the user with visual information on the status of the SFX engine including which sound effects models are currently invoked, the structure of these sound models, the settings of their parameters, and other information.

The sound models consist of the interface functions, the parameters for external control, private data for maintaining state while the process is suspended to share CPU time, indexes into the bank wavetables the model uses, and the event-generating code.

The sound models are arranged as an object-oriented class hierarchy, with many sound classes being derived directly from the base class. This structure is due to the fact that there are many attributes and methods common to all sounds (e.g. location, volume), while most other attributes are common to one model, or shared with other models that otherwise have little in common (e.g. surface characteristics of footsteps).

The models are based on some sound-generating phenomenon. Some examples of a phenomenon are footsteps, earthquake, running air-conditioner, bouncing ball, moving car, etc. The phenomenon can be virtually anything so long as there are some sounds with which it is associated. Indeed, the phenomenon need not even necessarily be a real-life phenomenon in the sense that it does not have to actually exist in the real world.

The sound modeling process begins by identifying the behavioral characteristics associated with the particular sound phenomenon which are relevant to the generation of sound. Behavioral characteristics can be

defined as the set of properties which a naive listener would perceive as distinguishing the sound effect from other sound effects, including those which define how it changes or evolves in response to different conditions impinging upon it.

For some of these conditions, it is useful to analyze the mechanics of how the sound is generated from the phenomenon. Using footsteps as an example, the sound being generated from footsteps results mainly from two separate events, the impact of the heel hitting a surface, then shortly after, the impact of the toe hitting a surface. For footsteps of a normal person, the totality of the sound generated results from the heel-toe action of one foot followed by the heel-toe action of the other foot, and so on. As one walks faster, the time interval between the sound produced from the heel-toe action of one foot and heel-toe action of the other foot decreases. However, it is important to also realize that the time interval between the sound produced from a heel and then a toe also decreases in some relationship to the heel-heel time interval.

Once the behavioral characteristics have been identified, some sound samples or procedurally generated sound is needed which will become the foundation for the many variations. The sample may be obtained either by recording a sample segment of actual sound found in a real-world phenomenon (or simply taking a segment from some existing pre-recording) or by producing a segment through well-known synthesis techniques, whichever is more convenient or desirable given the particular sound effect being modeled.

The choice of the length of the sound samples depends on a number of factors. As a general rule, the smaller the sample, the greater the

flexibility. On the flip side, the smaller the sample, the greater the labor and the harder it is to achieve realism. A good rule of thumb is to have a sample which is as long as possible without loss of useful flexibility, that is, where most of the perceptual range of sonic possibilities of the equivalent sound in real life can be achieved by varying the user parameters of the model. The choice of the sound samples also depends on the behavioral characteristics of the phenomenon to some extent, and also on the limitation of the parameters.

Once the behavioral characteristics have been analyzed and the samples obtained, it is necessary to select the user parameters, i.e., the parameters which are to be made available to a user of the model in order to control the sound effects. The parameters represent the various factors which need to be controlled in order to produce the modeled sounds. Although the parameters can be structured in a number of ways to effect a sound effect, for the preferred embodiment of the present invention, it is useful to view the parameter structure as having three layers, top, middle, and bottom.

The top layer consists of the user parameters which are the interface between the user and the sound effects system. The middle layer consists of the parameters employed by the SFX engine, or simply referred to as "engine parameters." The bottom layer consists of the synthesizer parameters which are well-known parameters found in any of the current sound or music synthesizers.

Each of the user parameters affects a combination of engine parameters and synthesizer parameters, though, in simpler cases, a user parameter may control only synthesizer parameters or engine parameters.

Any combination of engine and synthesizer parameters is theoretically possible; however, the way in which they are combined will depend on how the user parameter is defined in light of the behavioral characteristics of a particular phenomenon.

The user parameters are defined in terms of the desired sound effect. For instance, in the case of the footsteps, the user parameters can be location, walking speed, walking style, limp, weight, hardness, surface type, etc. Although these parameters can be defined in virtually any manner, it is often most useful if they directly reflect the behavioral characteristics of a phenomenon and the purpose for which the sound effect is being produced.

The middle layer parameters, or engine parameters, and the bottom layer parameters, or synthesizer parameters, work in combination to produce the sound effects as defined by the user parameters. The bottom layer parameters can include sound manipulation techniques such as volume control, pan, pitch, filter cutoff, filter Q, amplitude envelope, and many others which are well known to those skilled in the art.

When and how the middle and bottom layer parameters are combined is controlled by the SFX engine which takes into consideration the behavioral characteristics of a particular phenomenon. Essentially, the middle layer can be viewed as the layer which "models" the sound using the basic sound manipulation parameters provided by the bottom layer. Although the role of the middle layer parameters is complex, the parameters can broadly be classified as timing, selecting, and patterning.

The timing parameters basically control the length of the time intervals between triggering and stopping pieces of sound within a particular sound effect, and time intervals between other commands sent to the synthesizer.

The selecting parameters control which sound samples are selected at a given moment, including the order in which samples are selected. The patterning parameters control the relationships between these factors. By appropriately adjusting these three classes of engine parameter in combination with the synthesizer parameters, a large set of sound effects can be produced.

Although generally the behavioral characteristics will have some bearing on the choice of the synthesizer parameters to be used, there is no hard and fast rule as to how these parameters should be selected. Because sound is somewhat defined by human perception and also because there are many subtle variations, the study of the behavioral characteristics of a phenomenon may not always reveal sufficient information to determine how synthesizer parameters should be used for a given situation. Hence, some empirical experimentation may need to be performed.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is block diagram illustrating the preferred embodiment of the present system.

FIG. 2 is a functional block diagram illustrating the logical structure of the present system.

FIG. 3 is illustrates a graphic user interface showing user parameters for the sound effects footsteps where the user parameters are represented in the form of graphical sliders.

FIGS. 4A through 4E are flow diagrams illustrating the various steps the present system follows for various commands.

FIG. 5 illustrates the parameter structure employed by the preferred embodiment of the present system.

FIGS. 6 - 14 are tables charting the parameters used for actual sound models.

DETAILED DESCRIPTION OF THE INVENTION

SYSTEM CONFIGURATION

The overall system can be conceptualized as several layers, the top layer being applications that will use the sounds. The graphical user interface is just one such application (though a rather special one). The next layer is the collection of algorithmic sound models. These are objects that encapsulate data and describe the sound behavior. The models provide the interface which applications use to control the sounds by playing, stopping, and sending messages or by updating parameters.

Sound models generate commands and pass them at the proper time to the synthesizer. The synthesizer is the back-end where the sample-by-sample audio waveforms are produced, mixed, and sent to the audio output device.

FIG. 1 is a block diagram illustrating the main elements of the preferred embodiment of the present invention. A central processing unit 1 is connected to random-access memory (RAM) 2, one or more input devices

such as keyboard 3, mouse 4, joystick 5, MIDI controller 6; a visual display device 7; a sound synthesizer 8; and audio output system including digital-to-analog converter (DAC) 9, amplifier 10, loudspeaker or headphone 11 (or alternatively, a soundcard integrating some of these individual devices can be used); and a non-volatile storage device such a hard disk 12. These components play a supporting role to the central element of the invention, the interactive sound effects computer program (SFX program) which consists of a sound effects software engine (SFX engine) 13 and optionally a graphical user interface program (GUI) 14. In one mode of operation the GUI can be replaced by a controlling program 15 which replaces the GUI. Also present is an operating system program 16, such as the standard operating system of any personal computer system.

The CPU 1 executes the stored instructions of the programs in memory, sharing its processing power between the SFX engine 13, the GUI 14 or controlling program 15, the operating system 16 and possibly other programs according to a multitasking scheme such as the well-known time-slicing technique. Under command of the SFX engine, it delivers a stream of commands to the sound synthesizer 8, which produces digital audio in response to these commands. The output of the synthesizer is a digital audio signal which is converted to an analogue form by the digital to analogue converter (DAC) 9, then amplified and delivered to the user by means of the amplifier 10 and loudspeaker or headphones 11. Optionally the digital audio signal may also be delivered back to the CPU allowing it to be further processed or stored for later retrieval.

The hard disk or other nonvolatile storage 12 provides means to store indefinitely the following items:

- the SFX program itself including the data and instructions representing multiple sound effects models
- settings of parameters and other variable elements of the SFX program
- optionally, digital audio output from the synthesizer 8 under control of the SFX program.

The SFX engine 13 is controlled directly by means of the GUI 14, or from an external controlling program such as a computer game 15 or rarely by both at the same time. When under control of the GUI, the user effectively interacts directly with the SFX program, controlling it by means of one or more input devices such as an alphanumeric computer keyboard 3, a pointing device 4, a joystick 5, a specialized controller such as a slider bank or music keyboard connected by means such as the Musical Instrument Digital Interface (MIDI) standard 6, or other physical controlling means. In this mode of use, the GUI program 14 uses the display device 7 to provide to the user with visual information on the status of the SFX engine 13 including which sound effects models are currently invoked, the structure of these sound models, the settings of their parameters, and other information.

When the control is by means of a pointing device, the display device 7 also provides feedback to the user on the logical position of the pointing device in the usual manner. By observing the display 7 and/or listening to the audio output while manipulating the input devices 3 through 6, the user is able to alter sound effects until satisfied with the results. This mode of operation is designed to allow the user to create specific sound effects

according to his/her needs from the generic sound effects models of the SFX system, by selecting sound effects models, initiating or triggering them to produce audio output from the system, adjusting the parameters of the models, selecting elements of models, and other actions.

In the alternative mode of operation the SFX engine 13 is under the control of an external controlling program 15, such as a computer game, the program of a network-resident information site (website), a virtual reality program, a video editing program, a multimedia authoring tool, or any other program which requires sound effects. In this mode the user interacts with the controlling program 15 by means of the input devices 3 through 6 and the display device 7. The SFX engine 13 acts as a slave to the controlling program 15, producing sound effects under its control. This is achieved by allowing the controlling program to send data to the SFX engine 13, this data being interpreted by the SFX engine as controlling messages. In this mode of operation, the SFX engine will typically not be visible to the user on the display 7, and will be controllable by the user only indirectly via aspects of the controlling program which influence the SFX engine. The manner and degree of control which the user has over the SFX engine is entirely a function of the controlling program and is decided by the designer of the controlling program.

LOGICAL STRUCTURE

The logical structure of the present system is shown in Fig 2. The main elements are the SFX engine 1 which, as described above, may be

under control of the GUI 2 or, in the alternative mode of operation, under control of an external controlling program 3. Also shown is the synthesizer 4 which leads to the audio output system. These elements are the same as the corresponding elements of Fig 1, but are here shown in a way which highlights their logical interrelationships.

In the mode of operation where the invention is being used directly by a user, the user controls the system by means of the GUI 2, which acts to accept user input (such as keystrokes of the computer keyboard or movements of a pointing device) and to inform the user both of the status of the system and of the effect of his/her actions. User actions which affect the production of sound effects generate control messages which are sent from the GUI to the SFX Engine 1 in order to initiate, terminate, and control sound effects. These messages are in a format determined by the SFX Engine and known to the GUI. In response to these messages, the SFX engine 1 models the behavior of the currently-active sound effects and generates a stream of events or commands which are sent to the synthesizer 4, which in turn generates the audio output. Certain information affecting the manner of display to be used by the GUI 2 is contained within the SFX engine 1: for example the manner in which the control parameters of a sound effects model should be displayed varies from one model to another, and the information about the currently-active sound effects models is held by the SFX engine. Thus there is a need for information to be returned from the SFX engine to the GUI, and this is achieved by allowing the SFX engine to send display information to the GUI or allowing the GUI to elicit display information from the SFX engine.

In the alternative mode of operation where the SFX engine is being controlled from a program external to the invention, the user interacts with the external controlling program 3 in a manner which is completely independent of the invention. The controlling program 3 sends control messages to the SFX engine 1 in order to initiate, terminate, and control sound effects. These messages are in a format determined by the SFX Engine and known to the controlling program, and typically are similar to, or a subset of those used by the GUI in the first mode of operation described above. In response to these messages, the main purpose of the SFX engine 1 is to model the behavior of the currently-active sound effects and generate a stream of events or commands which are sent to the synthesizer 4, which in turn generates the audio output.

The main internal elements of the SFX engine 1 are a set of interactive sound effects models (SFX models) 5; an Application Programmer Interface (API) 7; A Message Processor 8; a Parameter Linker/Mapper 9; A Timing and Synthesizer Command Processor (TSCP) 10.

In the library or set of interactive sound effects models (SFX models) 5, each model consists of data and programmed instructions representing the sound characteristics and behavior of a sound effect, or a class of sound effects. These models may be invoked sequentially or simultaneously, so that the system is capable of producing sound effects in isolation or in combination, typically after an imperceptible or near-imperceptible delay (in so-called "real time"). Each SFX model is provided with one or more control parameters which may be used to alter the sound produced by the SFX model, and these control parameters may also be modified in real time to

produce audible changes in the output while the system is producing sound effects. In certain cases compound sound effects models may be made up of other sound effects models arranged in a hierarchy consisting of any number of levels, thus enabling arbitrarily complex models to be built from a number of simpler models.

The Application Programmer Interface (API) 7 receives data which is interpreted by the SFX engine as controlling messages, these messages arriving from either the GUI 2 or the external controlling program 3. The API decodes the messages in order to establish which type of message has been sent, and forwards the messages to the Message Processor 8.

The Message Processor 8 performs actions as directed by the controlling messages, including starting and stopping particular sound effects, loading and unloading sound effects models from RAM, applying the effect of modifications of control parameters to the SFX models, modifying settings of the SFX engine which influence its overall behavior, and otherwise controlling the SFX Engine.

A Parameter Linker/Mapper 9 provides a means of endowing SFX models with one or more alternative sets of control parameters or meta-parameters, where these meta-parameters are linked to the original control parameter set of the SFX model or to other meta-parameters in a hierarchy of parameters. It also provides means of applying mathematical transformations to the values of control parameters and meta-parameters. The Parameter Linker/Mapper is useful because the original control parameters of a particular SFX model are not necessarily the most appropriate or useful in every case, for example when the SFX engine is being controlled by an external controlling program 3 which has its own

design constraints, or when the SFX model forms part of a compound SFX model as described above.

The Timing and Synthesizer Command Processor (TSCP) 10 provides a number of functions related to timing and to the processing of events and other commands to be sent to the synthesizer 4. The invention is not restricted to any particular method of synthesis, and details of this element depend significantly on the type and design of the synthesizer. However two general functions may be identified:

1. The SFX engine operates by producing a stream of commands such as MIDI commands which are delivered to the synthesizer in order to produce sounds, and typically this process occurs in real time. Most synthesizers operate by producing or modifying the output sound at the moment an event or command is received. A simple implementation of the SFX engine might therefore produce synthesizer commands only at the moment they are required by the synthesizer, but this is liable to timing disruption because the CPU may be unable to process the complex command stream of multiple SFX models quickly enough to avoid audible disruption of the output sound. Hence a more sophisticated implementation can achieve greater consistency of timing by generating the commands a short interval ahead of the current time, queuing them in a mechanism such as a data buffer, and delivering them to the synthesizer at the appropriate time. The TSCP provides this function in such a way that the interval by which commands are generated ahead of the current time may be adjusted to an optimum value which may also be set differently for different SFX models. The optimum is a compromise between the need to avoid timing

disruption and the need to make the system responsive to changes in its control parameters.

2. If more than one SFX model is active, or if a single SFX model is complex, there is a need to produce multiple command streams which must be delivered to different channels of the synthesizer, where each synthesizer channel is set up to create different sound elements of the sound effects. In typical implementations these channels are a limited resource and must be managed carefully, for example allocated dynamically upon demand. The TSCP acts as a synthesis channel manager.

As stated above, one purpose of the hard disk or other nonvolatile storage (12 in Fig 1) is to provide a means to store indefinitely the settings of parameters and other variable elements of the SFX program. Such parameters and other elements may be saved while the system is in the mode where it is being controlled directly by a user using the GUI, then recalled when the system is in the alternative mode under control of an external controlling program. This allows a user to experiment directly with the parameters of the sound effects using the GUI, save the set of values of the parameters found to be most appropriate to the application, then recall this same set of values while the SFX engine is under control of the external controlling program in order to have the system produce an identical or near-identical sound effect. Saving and recalling a sound effect in this way differs from saving and recalling a digital audio signal of the sound effect in that it is entirely based on a model of the sound effect and may therefore by altered after it has been recalled by means of changing its parameters.

The sound models may be modeled closely on the physics or other behavior of real-world objects so that the model produces realistic sound, and responds in realistic and/or predictable ways to parameter changes. The sound effects models may be assigned a set of control parameters deemed most important or appropriate to the particular sound effect in question, these being closely related to the behavioral characteristics of the sound generating phenomenon being modeled. This set of parameters may include parameters unique to the particular model, parameters that are generic to sets of similar models, and parameters that are generic to all models of the system. For example a model of human footsteps might have a parameter for walking style which would be unique to this model, another parameter for walking speed which would be common to all human and animal footstep models, and other parameters such as volume or reverberation depth common to all models.

The system can include models which are programmed with realistic simulations of naturally-occurring sound-producing entities, other sound effects which are exaggerated in character for dramatic effect, and other sound effects of a purely imaginative nature which have no counterpart in the real world. In the case of realistic simulations and exaggerations of real sounds, the sound effects models may be modeled to any chosen degree of precision on the behavior of their naturally-occurring counterparts, so that the sound effects models will automatically provide accurate reproductions of the sounds, sound-sequences or other audible characteristics of their naturally-occurring counterparts.

The system can also support "Compound Sounds": these are sound models consisting of a hierarchy of other sound models with any number of

levels in the hierarchy. Typically they may represent an entire scene consisting of many sonic elements. At the top level the user can make changes to the whole scene (e.g., changing the overall volume), but control over individual elements is also possible, and these lower-level elements can optionally be isolated (listened to "solo") when making adjustments to them.

The system includes generic support for "parameter linking" in which parameters may be linked to combinations of other parameters according to mathematical relationships; this allows, for example, high-level parameters to be used to make broad sweeping changes in multiple lower-level parameters, or to apply scaling to other parameters, or to make complex sets of changes in several other parameters.

To make the sound as realistic as possible, the system can introduce fluctuations (typically of a random or semi-random nature) into the sounds produced in order to avoid exact repetition and achieve a natural effect. Techniques for introducing fluctuations include:

- -- Altering the timing of commands or events sent to the synthesizer
- -- Altering the values of parameters in the commands sent to the synthesizer
- -- If the synthesizer is one based on replaying samples, randomly selecting samples from a collection of similar but non-identical samples

The system generates the stream of commands to the synthesizer a short interval ahead of the current time, this interval being set such that it is long enough to overcome potentially-audible disruption of the sound output which would occur if time-critical commands were generated at the moment they are required, but short enough that the system responds to changes in its control parameters after an imperceptible or near-imperceptible delay.

The system provides two modes of triggering. In one mode, the sound effects, typically of a continuous, evolving, or repetitive nature, will - once started - run continuously until explicitly stopped. In the other mode, the sound effects, typically of a short, non-continuous nature, are triggered each time they are required, thus allowing precise synchronization with visual events in a computer game, film, video production, or animation.

The system includes generic sound effects models in which the behavior of a class of sound effects is encoded, and which provides a method by which a user of the system can create specific sound models by selecting options of the generic models, setting the values of variables of the generic models to specific values, and providing the synthesizer with its own samples.

SOUND EFFECTS SYNTHESIS TECHNIQUE

Sound Representation

The sound models consist of the interface functions, the parameters for external control, private data for maintaining state while the process is

suspended to share CPU time, indexes into the bank wavetables or synthesis data the model uses, and the event-generating code.

The sound models are arranged as an object-oriented class hierarchy, with many sound classes being derived directly from the base class. This structure is due to the fact that there are many attributes and methods common to all sounds (e.g. location, volume), while most other attributes are common to one model, or shared with other models that otherwise have little in common (e.g. surface characteristics of footsteps).

The sound models have a compute-ahead window of time which is the mechanism by which they share the CPU. This window can be different for different sound models, and is usually in the range of 100-300 milliseconds. The sound model process is called back at this rate, and computes all the events up to and slightly beyond the next expected callback time. The events are time-stamped with their desired output times, and sent to the output manager (for further details see FIGs 4A and 4B, and description provided below).

Several aspects of sound model representation have been developed as a direct result of how multimedia applications developers need to control them. The first is parameter presets. These come from the frequent need to use a model with several different distinct parameterizations. Rather than burden the developer with having to write code to explicitly change each of perhaps many parameters, parameters may be adjusted using the graphical user interface ("GUI") while monitoring their effect on the sound, then stored as presets which can be recalled by the application. Besides developer convenience, the advantage of switching parameter settings in the application this way within a single

instance of a model rather than using two different instances of the model, is that the event generating algorithm (consider footsteps) can continue without breaking stride.

Another representation issue is the need for two different and mutually exclusive methods of control over many sounds. Conceptually, there are two different kinds of parameters; those which the application will use to interact with the sound in real-time, and those which select a particular parameterization of the sound from the database. These two groups may be different in different contexts for the same sound.

Consider a footsteps model. If a virtual environment application is designed so that the view of the world is through the user's eyes, then one of the natural controls of the footsteps sound would be a rate parameter. The faster the user moves through a space, the faster the rate of the footsteps. If, however, the footstep sound is attached to visible feet, then the individual sounds obviously need to be synchronized to a graphical event. In the first case, there is no graphical event and it would be posing a significant burden on the application to have to time and send a message to the sound model for every step. In the second case, a rate parameter is meaningless.

The present system provides for either or both methods of control. If event-by-event control is needed, the model's standard play function is not invoked, but the object provides a messaging interface which is used instead. All the other support (e.g. statistical variability of successive sounds, parameter control) is still available. For sound models that use rate to control other attributes, a meaningful rate must be measured from the event triggers.

A more complex issue of control can be illustrated with an applause model which, for example, is to be controlled in real-time using a parameter for the number of clappers. The parameter would typically start at 0, be driven up to a level corresponding to how many people are in the virtual audience, remain at that level for some time, then gradually decay back to zero. However, for certain purposes, an application may not need such intimate control. It may be preferable to simply specify the number of people and an "enthusiasm" level (a "meta-time" parameter) that could in turn affect the temporal envelope of the "number of people" parameter. The application would only have to concern itself with the "enthusiasm" parameter when (or before) the applause sound is initiated. The two methods of control are mutually exclusive.

The applause example is different from the footsteps example because with footsteps, both types of control discussed (individual footsteps vs. rate) are real-time. The contrasting methods of control in the applause example are between a meta-time specification of a temporal trajectory, and real-time control of the trajectory. We believe that the most useful way to support these control choices is to record parameter trajectories created by the developer using the GUI, and then use the trajectories during playback after a trigger event from the application.

These control issues arise not from the process of modeling the sound itself, but rather from the contexts in which the models are embedded. The same issues must be considered for every sound model, but the implementation of the different control methods depends heavily on the sound representation.

Sound Modeling Process

Below is a description which provides the steps for the sound modeling process. These steps are generic in the sense that they can be used to produce a wide range of sound effects, and are not limited to producing only a particular set. However, so that these principles are more easily understood, various examples will be provided and, at times, discussed for illustration purposes.

The present method and system produce sound effects which simulate sounds associated with a certain phenomenon. Some examples of a phenomenon are footsteps, earthquake, running air-conditioner, bouncing ball, moving car, etc. The phenomenon can be virtually anything so long as there are some sounds with which it is associated. Indeed, the phenomenon need not even necessarily be a real-life phenomenon in the sense that it does not have to actually exist in the real world. For instance, the phenomenon could be a firing of a futuristic phaser gun. Although such a gun may not currently exist (hence the phenomenon cannot exist), this fact is irrelevant so long as there is some perception about what the sounds associated with the phenomenon might be like or what might be acceptable to the listeners. It is also useful to have some perception about how the sounds would vary depending on various hypothetical factors. For example, one may perceive the sound associated with the firing of the phaser gun to become louder and sharper as the phaser gun becomes more powerful, and for it to be followed by various kinds of "ricochet" depending on what type of object is struck by the gun's beam.

The sound modeling process begins by identifying the behavioral characteristics associated with the particular sound phenomenon which are relevant to the generation of sound. Behavioral characteristics can be defined as the set of properties which a naive listener would perceive as distinguishing the sound effect from other sound effects, including those which define how it changes or evolves in response to different conditions impinging upon it. In many cases, they bear a one-to-one correspondence to the terms a layman would use to describe the sound effect. In the case of sound effects which do not correspond to an object or phenomenon which actually exists in the real world, e.g., phaser gun as mentioned above, the behavioral characteristics are properties which a naive listener might expect such an object or phenomenon to possess if it did exist.

For instance, in the case of footsteps, the behavioral characteristics would include things such as speed, degree of limp and stagger, weight (of the person producing the footsteps), surface type (e.g., cement, grass, mud), location (i.e., position relative to the listener), surrounding acoustic, etc. It can be easily appreciated that these characteristics define the sound for a particular set of conditions. For instance, the sounds produced from footsteps from a mad dash would be different from those produced in a casual stroll; footsteps on hard marble would sound differently than footsteps on wet mud.

For some of these conditions, it is useful to analyze the mechanics of how the sound is generated from the phenomenon. Once again using footsteps as an example, the sound being generated from footsteps results mainly from two separate events, the impact of the heel hitting a surface, then shortly after, the impact of the toe hitting a surface. For footsteps of a

normal person, the totality of the sound generated results from the heel-toe action of one foot followed by the heel-toe action of the other foot, and so on. As one walks faster, the time interval between the sound produced from the heel-toe action of one foot and heel-toe action of the other foot decreases. However, it is important to also realize that the time interval between the sound produced from a heel and then a toe also decreases in some relationship to the heel-heel time interval. At some point, the sound produced from the heel and the sound produced from the toe actually overlap and it becomes difficult to distinguish the sounds as being separate and distinct.

In addition, the heel-to-toe time is affected by another parameter. When marching, the leg falls rapidly and perpendicular to the ground, and thus, the heel-to-toe time is very short. In contrast, a long stride produces a long heel-to-toe time because the heel touches the ground while the leg is far from perpendicular and the toe has a relatively long distance to travel before it touches the ground. Thus, in the present method of modeling, the heel-to-toe time is the net result of both walking speed and "walk style" (march versus long stride). The general principle is that the internal parameters of the model may be influenced by many of the external or "user" parameters in mathematical relationships of arbitrary complexity.

In certain cases, this knowledge about the mechanics of sound generation is important for two main reasons when attempting sound modeling. First, it allows one to vary the correct set of parameters so as to produce the most realistic sound effect. For instance, in the footstep example given above, the resulting sound effect would not sound very realistic had someone varied the time interval between the sound produced

from one heel-toe action to another without also proportionately varying the time interval between the heel and the toe. The second main reason for analyzing the mechanics is that it allows one some notion of the size and type of sound sample that is needed. For instance, again using the footstep example above, it is important to have independent control of the heel sound and the toe sound individually, and therefore, a separate sample for each of the sounds is needed; it is not enough to have a sample of the heel-toe sound as a grouped pair.

Of course, there is a rather large range of behavioral characteristics of any particular phenomenon, and the choice of selection and the extent of the analysis of these behavioral characteristics depend largely upon the potential uses of the sound effects, the nature of the sound-producing phenomenon, and degree of realism desired. However, it is generally true that some identification and understanding of the behavioral characteristics of any phenomenon is required to properly model a sound effect.

Once the behavioral characteristics have been identified, some sound samples or other procedurally generated sound is needed which will become the foundation for the many variations. The sample may be obtained either by recording a sample segment of actual sound found in a real-world phenomenon (or simply taking a segment from some existing pre-recording) or by producing a segment through well-known synthesis techniques, whichever is more convenient or desirable given the particular sound effect being modeled. For instance, in a case where a sound of a phaser gun is being modeled, it may be more convenient to simply synthesize a sample, given than no such phaser gun actually exists in the real-world. In the case of footsteps, however, it would in most cases be

easier to simply record the sound produced from actual footsteps, or to record the individual elements of a footstep (i.e., heel-tap and toe-tap recorded separately), or to record a simulation of these elements (e.g., by tapping together different types of hard objects until the desired sound is achieved). When recording a sample for use in this type of model, it is generally better to isolate the sound, that is, to prevent the inclusion of sounds which are not related to the particular phenomenon at hand.

The choice of the length of the sound samples depends on a number of factors. As a general rule, the smaller the sample, the greater the flexibility. On the flip side, the smaller the sample, the greater the labor and the harder it is to achieve realism. A good rule of thumb is to have a sample which is as long as possible without loss of useful flexibility, that is, where most of the perceptual range of sonic possibilities of the equivalent sound in real life can be achieved by varying the user parameters of the model. For instance, in the case of the footsteps, if one were to want to produce footsteps of different speeds, it would be necessary to obtain a set of samples including heel sounds and toe sounds, for the reasons provided above. However, this does not always mean that one needs to record the two sounds separately since the current editing techniques allow for splicing and other forms of editing to separate a single recording into multiple samples. But the splicing technique may be difficult or impossible for cases where the sounds overlap.

The choice of the sound samples also depends on the behavioral characteristics of the phenomenon to some extent, and also on the limitation of the parameters (parameters are discussed in detail below). Using the footsteps example once again, it should be noted that some sound effects do

not require additional samples while some do. For instance, to vary the style of a walk, only the timing needs to be varied, and hence, this can be done with any existing sample. However, to vary the surface on which the footsteps are made, it is easier to simply obtain a sample of footsteps on each of the surfaces rather than attempting to manipulate an existing sample to simulate the effect. For example, it would not be easy to produce a sound of a footstep on a soft muddy surface using only a sample of footsteps on a concrete surface. How many samples are needed for a given phenomenon, of course, depends on the scope and the range of sound effects one desires, and varies greatly from one sound effect to another.

In many cases, multiple similar, but non-identical, samples are collected. This has two purposes. First, it provides a means of simulating the subtle, ever-changing nature of real-world sounds. For example, no two footsteps are identical in real life, and a model which produces two identical footsteps in succession is immediately perceived as artificial. With several samples to choose from and a rule which selects randomly from a set of similar samples (typically also excluding the same one being triggered twice in immediate succession), much of the naturalness may be simulated.

The second reason for collecting multiple samples is that a continuous spectrum can often be simulated by collecting points along the spectrum. For example, although there is no known synthesis or sound processing technique for transforming a quiet chuckle into a hearty laugh (or vice versa), a "strength of laughter" parameter may be constructed by collecting a set of laugh samples at different "degrees of hilarity", then selecting individual samples according to the setting of the "strength of

laughter" parameter. Typically, this technique is combined with the random selection described above.

Sound-Modeling Parameter Structure

Once the behavioral characteristics have been analyzed and the samples obtained, it is necessary to select the user parameters, i.e., the parameters which are to be made available to a user of the model in order to control the sound effects. The parameters represent the various factors which need to be controlled in order to produce the modeled sounds. Although the parameters can be structured in a number of ways to effect a sound effect, for the preferred embodiment of the present invention, it is useful to view the parameter structure as illustrated in FIG. 5.

In referring to FIG. 5, the top layer consists of the user parameters which are the interface between the user and the sound effects system. The middle layer consists of the parameters employed by the SFX engine, or simply referred to as "engine parameters." The bottom layer consists of the synthesizer parameters which are well-known parameters found in any of the current sound or music synthesizers.

As the arrows indicate, in general, each of the user parameters affects a combination of engine parameters and synthesizer parameters, though, in simpler cases, a user parameter may only control only synthesizer parameters or engine parameters. Any combination of engine and synthesizer parameters is theoretically possible; however, the way in which they are combined will depend on how the user parameter is defined in light

of the behavioral characteristics of a particular phenomenon, as shall be explained in detail below.

The user parameters are defined in terms of the desired sound effect. For instance, in the case of the footsteps, the user parameters can be location, walking speed, walking style, limp, weight, hardness, surface type, etc. Although these parameters can be defined in virtually any manner, it is often most useful if they directly reflect the behavioral characteristics of a phenomenon and the purpose for which the sound effect is being produced. In many cases, they are the obvious, easily-understood parameters that a layman might use to describe the sound. For example, while a user parameter such as surface type might be a useful parameter for the phenomenon footsteps, it probably would not be useful for a phenomenon such as earthquake, given that surface type probably has no meaning in the context of an earthquake.

The user parameters can be represented in a number of ways so as to give control access to the user. However, in the preferred embodiment, it is represented in the form of "sliders" on a graphic user interface (GUI), FIG. 3, where the user can slide the slider bar to control the magnitude of the effect. For instance, for the speed slider for the phenomenon footsteps, the walking speed is increased as the slider is moved to the right. For the limp slider, the amount of limp in the walk is increased as the slider is moved. To combine the effects, several user parameters can be invoked at once. For instance, by invoking both the speed slider and the limp slider, one can achieve any combination of limp and speed. Some combinations are obviously not desirable, though may be possible. For instance, one probably would not combine the surface type "metal" with "marble". In contrast, "leaves" might

well be combined with "dirt" to achieve an effect of footsteps on leaves over dirt.

The middle layer parameters, or engine parameters, and the bottom layer parameters, or synthesizer parameters, work in combination to produce the sound effects as defined by the user parameters. The bottom layer parameters can include sound manipulation techniques such as volume control, pan, pitch, filter cutoff, filter Q, amplitude envelope, and many others which are well known to those skilled in the art.

When and how the middle and bottom layer parameters are combined is controlled by the SFX engine which takes into consideration the behavioral characteristics of a particular phenomenon. Essentially, the middle layer can be viewed as the layer which "models" the sound using the basic sound manipulation parameters provided by the bottom layer. Although the role of the middle layer parameters is complex, the parameters can broadly be classified as timing, selecting, and patterning. Although these parameters are defined here as being separate and distinct, it should understood by those skilled in the art that these parameter representations are conceptual tools to illustrate the sound modeling process or techniques employed by the SFX engine and need not necessarily exist as separate and distinct components in the present sound effects system.

Now in describing the role of each class of parameter individually, timing parameters basically control the length of the time intervals between triggering and stopping pieces of sound within a particular sound effect, and time intervals between other commands sent to the synthesizer. The selecting parameters control which sound samples are selected at a given moment, including the order in which samples are selected. The patterning

parameters control the relationships between these factors. By appropriately adjusting these three classes of engine parameter in combination with the synthesizer parameters, a large set of sound effects can be produced. The role and the effect of each of these parameters will become clearer in the examples as provided below.

Referring to the footsteps example once again, described above were the behavioral characteristics of footsteps in relation to speed. It was explained that as the speed increases, the time interval between one heel-toe action to another decreases, as well as the time interval between the heel and the toe. Here, the user parameter (top layer) is speed. As the user adjusts the speed slider to increase speed, the time parameter is made to decrease the time interval between one heel-toe action to another, as well as the time interval between the heel and the toe. This timing is also affected by the "style" parameter as described above. However, the pattern or behavior of the footsteps does not change as speed and style are altered. A heel sound is always followed by a toe sound, etc.

If, however, the sound effect were that of a moving horse, then the behavioral characteristics are more complicated, and hence, additional parameters need to be involved. For example, consider the user parameter "speed" for the sound of horse's hooves. As the speed increases, it is clear that the timing parameter needs to be adjusted such that the mean of the time intervals between the events becomes shorter, reflecting the fact that the time intervals between impacting of the hooves to a given surface become shorter on average. But, in addition, the patterning and ordering of the events change as the horse switches between walking, trotting, cantering

and galloping. The exact pattern, of course, needs to be determined empirically using the behavioral characteristics of an actual horse.

As the last example, if for the footsteps phenomenon, the user parameter were surface type, then the only class of engine parameters affected are those concerned with selection, since the timing and patterning aspects do not change. Here, depending on what surface or combination of surfaces is chosen different sets of samples will be selected, but the timing and patterning do not change.

For some of these examples, there may be instances where synthesizer parameters will have to be invoked either in isolation or in combination with engine parameters. For instance, still using the footsteps example, the synthesizer parameters, pitch, volume, etc., need to be controlled in response to the user parameter, weight (of the person making the footsteps), since typically a heavier person would produce footsteps which are deeper in pitch, louder, etc. (though this may not always be true in real life). Although generally, the behavior characteristics will have some bearing on the choice of the synthesizer parameters to be used, there is no hard and fast rule as to how these parameters should be selected. Because sound is somewhat defined by human perception and also because there are many subtle variations, the study of the behavioral characteristics of a phenomenon may not always reveal sufficient information to determine how synthesizer parameters should be used for a given situation. For instance, it has been found that to produce the best sounding effect for horse's hooves, it is helpful change the tonal quality as the speed increases. The relationship between tone and speed is not necessarily obvious. Hence, some empirical experimentation may need to be performed.

To further illustrate the principles provided above, the FIGs. 6 through 14 are tables charting the various parameters that were used for some actual sound models. Taking the table in FIG. 6 as an illustrative example, the first column lists the user parameters plus "random fluctuations" (see above for description for "random fluctuations"). The subsequent columns have a heading at the top showing the engine and synthesizer parameters, the engine parameters comprising the first three columns subsequent to the user parameter column. The "X" in a box indicates that the parameter in that column was used for the sound modeling for the user parameter found in that particular row.

These tables show how changes in user parameters affect a) aspects of event patterning, and b) different synthesizer parameters. Also (on the first row of each table) they show which parameters are affected by the small random fluctuations that are introduced to provide naturalness and variety in the sound effect, even where there is no change in a user parameter.

In FIG. 8, the user parameters, Break, Clutch, and Gas Pedal, control two "internal" variables, car speed and engine speed. The two internal variables are governed by a 2-D differential equation with the Pedal settings as inputs. The car speed and engine speed in turn control the synthesizer event generation. The engine speed controls the firing of pistons, each firing is a separately triggered event (many hundreds per second). The car speed controls the "rumble" of the car strolling along the road.

In FIG. 14, the wind sound model consists of a very small number of events. Most of the user parameters (and the random fluctuations) affect the same set of synthesis parameters, i.e., volume, pitch, filter, etc., but they affect them in different ways. For instance, "Strength" controls the mean

value of the parameters (stronger wind has higher volume, pitch, filter Q, etc). The "Width of Variation" controls the deviation from the mean (of the same parameters) and "Gustiness" controls the rate of change of the parameters. "Wind Strength" also controls the number of layers (e.g., number of "whistles") in the sound.

Development environment

The development environment for embedding sound models into applications needs to be richer than the traditional one where a recorded sound is selected from a CD, loaded into the environment, played and stopped. This is partly because sound selection involves more than selecting a track, but rather involves adjusting parameters. There is thus an important role to be played by an interactive environment which is separate from the target application.

The developer's tool consists of two parts; the GUI, Figure 3, and the SFX engine. Using the GUI, the developer browses the database of sound models, chooses the sound models that will be used in an application, and adjusts the sound parameters while listening to the sound in real-time. There is no code-compile-listen cycle; sounds are chosen and adjusted by ear. The parameters and their names are high-level intuitive, and related closely to behavioral characteristics of the real-world sound which is being modeled (e.g., "roughness" rather than "AM depth and regularity"). Once the sound has been chosen and parameterized, the developer creates file names where the customized sound models (i.e., as set by the developer using the user parameters) will be stored. To then play and control the

sound models from within the application, the API calls described below are used.

Application Programmer Interface (API)

Functions are provided to the application for initializing the audio engine, loading sounds, creating a map between application variables and sound parameters, playing and stopping sounds, and sending user parameters and messages to sounds.

Sounds can be loaded and unloaded from memory as needed. A call to *PLAY*, FIG. 4A, will initiate the playing of the sound over the computers sound system, and a call to *STOP*, FIG. 4D, will cause the sound to cease playing. After a sound has been stopped, it can be played again, unless a call to unload has been made, in which case the sound model is removed from memory.

If no interactive control of the sound is desired, the *PLAY* and *STOP* are the only routines that need to be called. Thus, sound models can be used exactly as digital audio files have always been used by applications. Developers are confronted with nothing that forces them to deal with the fact that sounds are being generated algorithmically in real-time rather than being played from a file, other than that they may notice that sounds are (in general) more natural and pleasing because they are not identical each time they are played.

If interactive control of sound model parameters is desired, then a call to an API function called *SetParam* (FIG. 4C) accomplishes the task. Although some user parameters are common to all sounds (e.g. relative

location of sound emitter and listener), and others are unique to a particular sound, all user parameters are updated through the same base sound class function. The GUI additionally plays a sound documentation role here in that the various parameter ID's a sound accepts via the API are the same as the names displayed along with the sliders in the GUI.

Typically, the application will control a sound using an application variable that has a range of possible values determined by its role in the application. On the other hand, the sound model user parameter has its own range of possible values determined by how the user parameter is used within the sound object. The range of the application variable can be automatically mapped on to the range of the user parameter with a call to a function called *SetParamMap*. This call only needs to be done once for a given user parameter, after which any call to *SetParam* will automatically perform the proper mapping. Sound locations are similarly controlled by a single call to a "mapping" function to establish the units, and "set" functions to update the location of the sound emitter and receiver.

Finally, some sounds take trigger messages that can be sent from the application, FIG. 4E. Again, while the messages a sound accepts are unique, all call the same base sound method (which simply does nothing in response to messages a particular sound does not handle). These messages usually have the effect of immediately triggering some element of a sound, such as a train whistle when a steam train sound is playing. This messaging method is also the means by which event-by-event control is implemented (e.g., triggering individual footsteps), and the way parameter presets are switched.

Details of Implementation

To illustrate the implementation details, FIG. 4A shows what happens when play is initiated, either in response to a user interacting through the GUI, or a message arriving from a game or other application program. After a number of initialization tasks, including a request for the necessary synthesizer channels, the system picks up the current settings of the user parameters. It then generates a set of commands such as MIDI messages which are sent to the synthesizer. It generates these a short way into the future - a time [X + Safety Margin] which is long enough to ensure that delays in the system will never result in gaps in the audio output, but short enough that changes in user parameters cause results which are perceived as almost immediate by the user. The system then initiates a system timer which will callback (i.e. return control to) the SFX engine after time X. In the meantime the computer is free to process other programs such as a game which the SFX Engine may be embedded into. Time X is typically of the order of 10 - 100 mS, and the Safety Margin is typically of the order of 5 - 10 mS.

FIG. 4B shows what happens after time X. At this point the buffered stream of synthesizer commands generated as described above is almost empty (commands corresponding to an amount of time equal to the Safety Margin is left before it underruns). The system first processes any changes in parameters since the last call -these may include the internal results of changes in any of the user parameters and (in some models) changes due to the progress of time. Following this, it makes small random changes in certain internal parameters in order to introduce subtle fluctuations in the

sound - this is required for a natural or pleasing effect. It then generates another stream of commands for the following period X, and arranges for another callback after that time. This generate-callback cycle will continue until the system is stopped.

FIG. 4C shows what happens when one of the user parameters is changed. As described above, the system includes a sophisticated mapping scheme where a change in a user parameter can cause changes in any number of internal parameters of the model. (This is required in order to ensure that the user parameter behaves in a way which corresponds well with the behavioral characteristics of the phenomenon being modeled.) This mapping scheme also provides for both linear and non-linear transformations of the user parameter value, and these transformations may be distinct for each of the possibly-many mappings of user parameter to internal parameters of the model. After mapping, the parameter changes may also result in the immediate sending of one or more synthesizer commands. In other cases these parameter changes are acted upon at the next callback (see above).

FIG. 4D shows what happens when a stop message is received, in response either to a user interacting through the GUI, or a message arriving from a game or other application program. The system first cancels the pending callback. It then generates synthesizer commands into the future as required to produce the desired ending to the sound (for some sounds this may be an abrupt stop, in others it must be more gentle for a natural or pleasing effect; in some cases, it may involve modifications of synthesizer parameters). Finally it releases the synthesizer channels which were granted when the sound was started, and performs a number of other

cleanup functions - for example resetting the synthesizer channels and variables of the system ready for re-use by a subsequent Play message.

Flowchart 4E shows what happens when a trigger message is received, either in response to a user interacting through the GUI, or a message arriving from a game or other application program. Trigger messages are designed to produce a sound effects (or parts of sound effects) of finite length (typically a few seconds or less) which stop of their own accord after initiation. Thus they act basically like a Play with an automatic Stop, and this flowchart should be self-explanatory in the light of the descriptions above.

Process Management

The process management is one function of the TSCP introduced above and shown in FIG. 2. It mediates both commands destined for the synthesizer and the process callback requests. The interrupts that system timers generate for their decrement and test cycle make them an expensive resource. If each of possibly many concurrent sound process were requesting callbacks from the system directly, servicing the interrupts could significantly degrade performance. Instead, a time sorted queue of processes awaiting callbacks is maintained by the process manager; and a system timer is engaged only for the next sound process to run.

A similar mechanism is used for commands sent to the process manager time-stamped for output to the synthesizer. Although each sound process generates commands in chronological order, when many processes are running concurrently, the process manager inserts

commands in a sorted queue, and sets up a timer only for the next one scheduled to go out.

Some commands, such as interactive controller messages, are serviced immediately rather than queued. The back-end synthesizer uses these commands to influence any new events that the process manager sends, even if the events were queued before the "crash through" controller update. Thus, user parameters have a latency that can be as great as the callback period for a sound process, but parameters that control the synthesis have a shorter control-rate latency.

The process manager is also responsible for allocating synthesizer resources to sound processes and interfacing with specific back-end synthesizer API's.

The present invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The presently disclosed embodiments are, therefore, to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are, therefore, to be embraced therein.

CLAIMS

We Claim:

1. A sound effects system for producing sound effects modeled after a sound-generating phenomenon, said phenomenon having a set of behavioral characteristics, said system comprising:

a computing apparatus for processing a model for each of the sound effects, said computing apparatus including a central processing unit, memory, and an input device;

a synthesis means for synthesising sounds, said synthesis means controlled by said computing apparatus;

a sound effects model, said model being a representation of the behavioral characteristics of the corresponding sound-generating phenomenon;

an output means for outputting the sound effects;

a command generation means for generating a stream of synthesizer commands derived from said sound representation, said commands being communicated to said synthesis means such that said sound synthesis means produces audio output which is representative of the behavioral characteristics of said sound-generating phenomenon, said synthesizer commands including commands to initiate and terminate sound events and commands to control synthesizer parameters; and

a means for controlling a user parameter by which said audio output may be changed in ways which relate to said behavioral characteristics of said sound-generating phenomenon;

whereby said system produces a wide range of realistic sound effects and their variations in an efficient manner.

2. The sound effects system as recited in Claim 1 further including a graphical user interface, said graphical user interface allowing a user to control said sound effects.

3. The sound effects system as recited in Claim 1 further including an application programmer interface, said application programmer interface allowing tasks or processes executing on said computing apparatus to control said sound effects.

4. The sound effects system as recited in Claim 2, wherein the sound effects corresponding to different sound-generating phenomena are controlled by a common graphical user interface.

5. The sound effects system as recited in Claim 3, wherein the sound effects corresponding to different sound-generating phenomena are controlled by a common application programmer interface.

6. The sound effects system as recited in Claim 1 wherein said input device is selected from the group consisting of keyboards, computer pointing devices, trackballs, joysticks, buttons, knobs, sliders and communication ports.

7. The sound effects system as recited in Claim 1, wherein said sound events are derived from replaying segments of recorded audio data.

8. The sound effects system as recited in Claim 1, wherein said sound events are derived from the procedural generation of audio data.

9. The sound effects system as recited in Claim 1, wherein said sound synthesis means is a peripheral communicably connected to the computing apparatus.

10. The sound effects system as recited in Claim 1, wherein said sound synthesis means is a process executing on the central processing unit of the computing apparatus.

11. The sound effects system as recited in Claim 1, wherein said sound effects model includes mechanisms for introducing fluctuations into the sound output whereby said fluctuations substantially imitate the fluctuations occuring in non-synthetic sounds, thus making the sound output more realistic.

12. The sound effects system as recited in Claim 11 wherein said mechanisms include a means for altering the timing of synthesizer commands including commands which initiate and terminate sound events and commands which control synthesizer parameters.

13. The sound effects system as recited in Claim 11 wherein said mechanisms include a means for randomly selecting sound events from a collection of similar but non-identical sound events.

14. The sound effects system as recited in Claim 11 wherein said mechanisms include a means for altering the values of synthesizer parameters.

15. The sound effects system as recited in Claim 1 further comprising:

a means for arranging a plurality of similar but non-identical sound events in an order, the order being determined by strength of properties of said events, said properties including acoustic properties and behavioral characteristics;

a means for selectively initiating and terminating said sound events according to a relationship between a value of said user parameter and said strength of said properties;

whereby said user parameter will substantially simulate the effect of controlling a behavioural characteristic of the sound effects along a continuous spectrum.

16. The sound effects system as recited in Claim 1, wherein said synthesizer commands are generated ahead of the current time by a predetermined time interval and are held in a buffer until they are sent to said sound synthesis means, said predetermined time interval being long enough to offset delays while said commands are generated but short enough that said system responds to changes in said user parameters after a substantially imperceptible delay.

17. The sound effects system as recited in Claim 1, wherein said sound effects model includes a plurality of other sound effects models arranged in a hierarchy of levels.

18. The sound effects system as recited in Claim 1 wherein said synthesizer parameters are selected from the following group: volume, pan, pitch, filter cutoff, filter Q, amplitude envelope.

19. The sound effects system as recited in Claim 1 wherein said stream of synthesizer commands is generated by a process governed by engine parameters, said engine parameters controlling:

selection of synthesizer commands including commands which initiate and terminate sound events and commands which control synthesizer parameters;

timing of synthesizer commands including commands which initiate and terminate sound events and commands which control synthesizer parameters; and

patterning of synthesizer commands including commands which initiate and terminate sound events and commands which control synthesizer parameters.

20. The sound effects system as recited in Claim 19 wherein first parameters are mapped to second parameters so that any alteration in a value of one of said first parameters will cause an alteration in a value of said second parameters, said first parameters being selected from the group consisting of user parameters and engine parameters, said second parameters being selected from a group comprising user parameters, engine parameters and synthesizer parameters.

21. The sound effects system as recited in Claim 20 wherein a mapping relationship between first parameters and second parameters includes transformation selected from a group comprising linear transformations and non-linear transformations.

22. The sound effects system as recited in Claim 20 wherein a mapping relationship exists at a plurality of levels such that a second parameter at one level is also a first parameter at a next lower level.

23. The sound effects system as recited in Claim 20 wherein a first parameter and a corresponding second parameter belong to different sound models.

24. The sound effects system as recited in Claim 1 further including a mode of operation in which a sound effect is initiated each time a particular stimulus is received by said system and within a substantially imperceptible time of receiving said particular stimulus, said sound effect being produced by a communication of a timed stream of synthesizer commands to said sound synthesis means, said particular stimulus

arising from a source selected from a group comprising said input device of said computing apparatus and an external computer program, whereby said sound effect may be synchronised with events occuring in external media, said external media including motion video, animation, music, sound productions, computer games, computer graphical presentations, multimedia productions, lighting productions, theatrical productions and interactive physical environments.

25. The sound effects system as recited in Claim 1 further including a recording means for registering one or more parameter trajectories and a playback means for replaying said parameter trajectories at a later time, said parameter trajectories comprising a timed sequence of values of said user parameters.

26. The sound effects system as recited in Claim 1, further comprising:

a selection means for allowing a user of said system to select behavioral options of a generic sound effects model from a plurality of behavioral options in order to limit the behavioural characteristics of said generic sound effects model to the behavioral characteristics of a specific sound-generating phenomenon;

a variable-setting means to allow a user of said system to assign values to predetermined variables of the generic sound effects model in order to limit the behavioural characteristics of said generic sound effects model to the behavioural characteristics of a specific sound generating phenomenon;

a storage means for saving values of said options and said variables for later recall,

whereby a user of said system may create specific sound effects models derived from said generic sound effects models.

27. The sound effects system as recited in Claim 1 further comprising

an audio data import means for a user of said system to supply said sound synthesis means with additional audio data; and

an audio data storage means for saving said audio data for later recall,

whereby a user of said system may create specific sound effects models which combine the behaviour of said generic sound effects models with alternative audio data.

28. The sound effects system as recited in Claim 1 wherein one of said sound-generating phenomena is footsteps.

29. The sound effects system as recited in Claim 1 wherein one of said sound-generating phenomena is a firing of a gun.

30. The sound effects system as recited in Claim 1 wherein one of said sound-generating phenomena is a moving vehicle.

31. The sound effects system as recited in Claim 1 wherein one of said sound-generating phenomena is applause.

32. The sound effects system as recited in Claim 1 wherein one of said sound-generating phenomena is a trickling of a brook.

33. The sound effects system as recited in Claim 1 wherein one of of said sound-generating phenomena is a droning of a machine.

34. The sound effects system as recited in Claim 1, wherein one of said sound-generating phenomena is laughter.

35. The sound effects system as recited in Claim 1, wherein one of said sound-generating phenomena is a crowd cheering.

36. The sound effects system as recited in Claim 1, wherein one of said sound-generating phenomena is wind.

37. The sound effects system as recited in Claim 1, wherein one of said sound-generating phenomena is a non-real-world phenomenon.

38. A method of producing sound effects modeled after a sound-generating phenomenon, said phenomenon having a set of behavioral characteristics, said system comprising:

identifying a set of behavioral characteristics for said sound-generating phenomenon;

constructing a sound effects model, said model being a representation of the behavioral characteristics of the corresponding sound-generating phenomenon;

storing said model in a computing apparatus which is communicably connected to a sound synthesizer;

generating a stream of synthesizer commands derived from said sound representation, said commands being communicated to said synthesizer such that said synthesizer produces audio output which is representative of the behavioral characteristics of said sound-generating phenomenon, said synthesizer commands including commands to initiate and terminate sound events and commands to control synthesizer parameters;

providing the model with at least one user parameter, said user parameter corresponds to the behavioral characteristics of said sound-generating phenomena; and

manipulating said user parameter to produce variations of the sounds effects corresponding to said model.

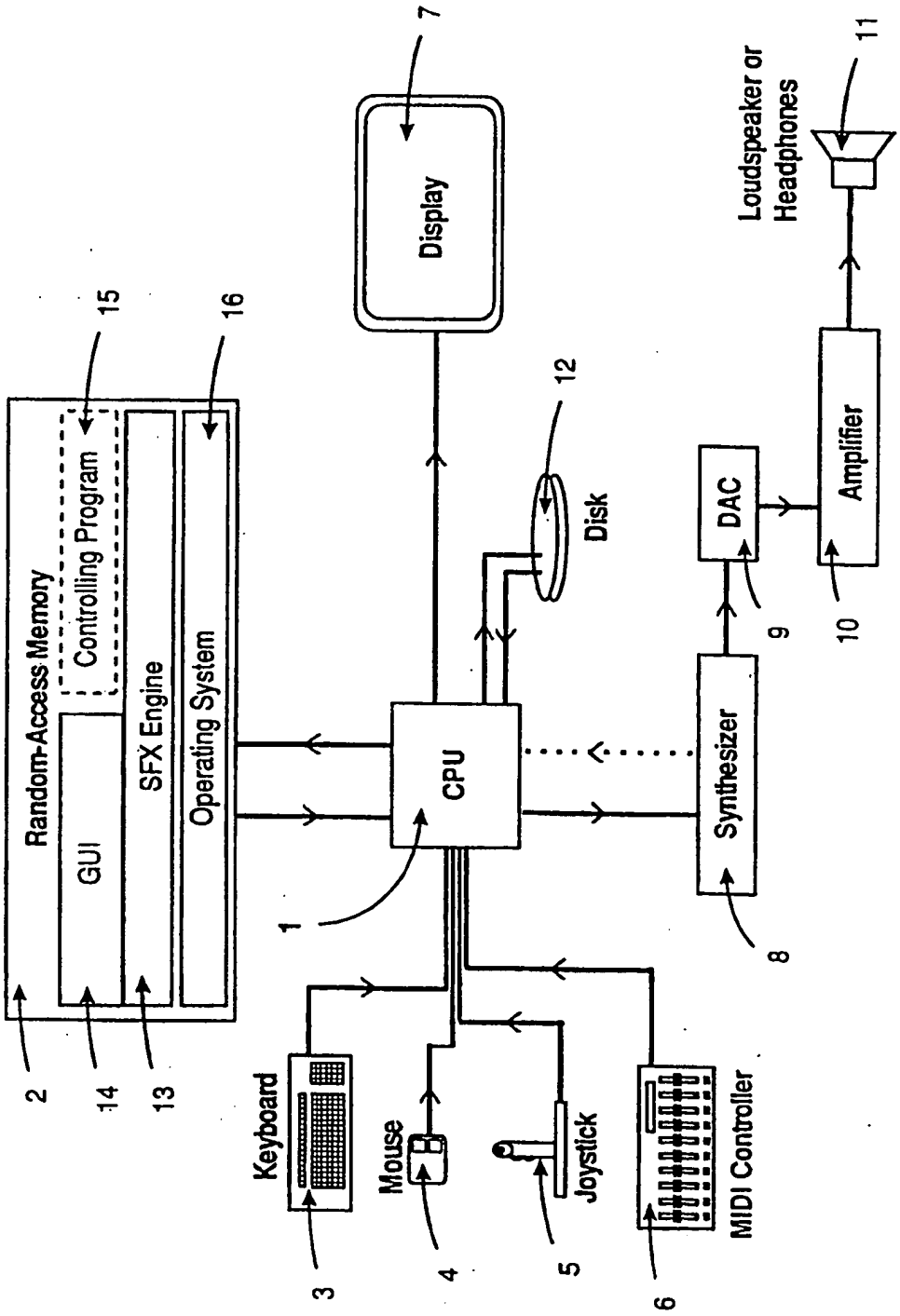

Random-Access Memory
Controlling Program
GUI
SFX Engine
Operating System
15
16
2
14
13
CPU
1
Display
7
Disk
12
Keyboard
3
Mouse
4
Joystick
5
MIDI Controller
6
Synthesizer
8
DAC
9
Amplifier
10
Loudspeaker or Headphones
11

FIG. 1

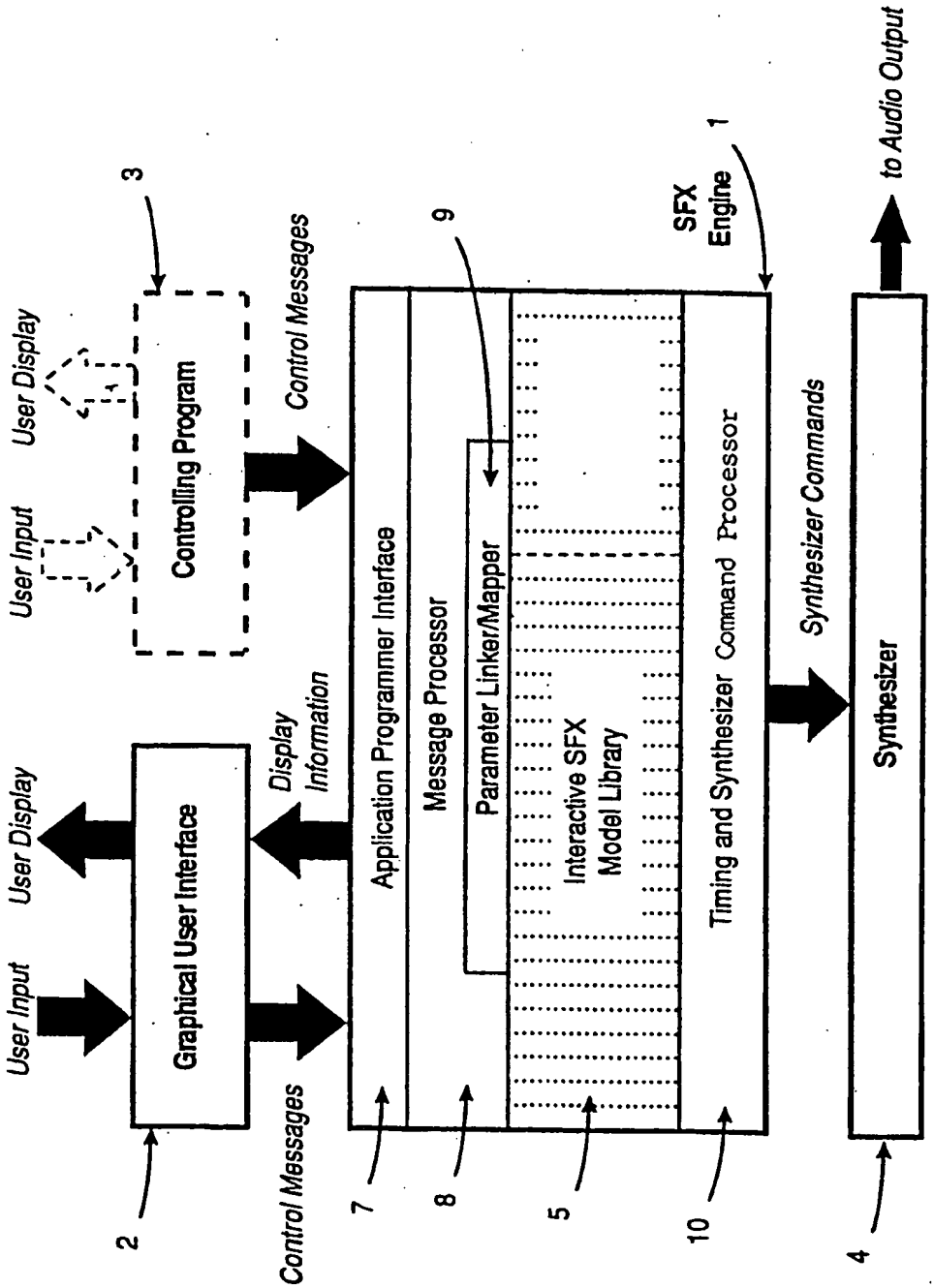

User Input
User Display
Graphical User Interface
2
Controlling Program
3
User Input
User Display
Control Messages
Display Information
Control Messages
Application Programmer Interface
7
Message Processor
8
Parameter Linker/Mapper
9
Interactive SFX Model Library
5
Timing and Synthesizer Command Processor
10
SFX Engine
1
Synthesizer Commands
Synthesizer
4
to Audio Output

FIG. 2

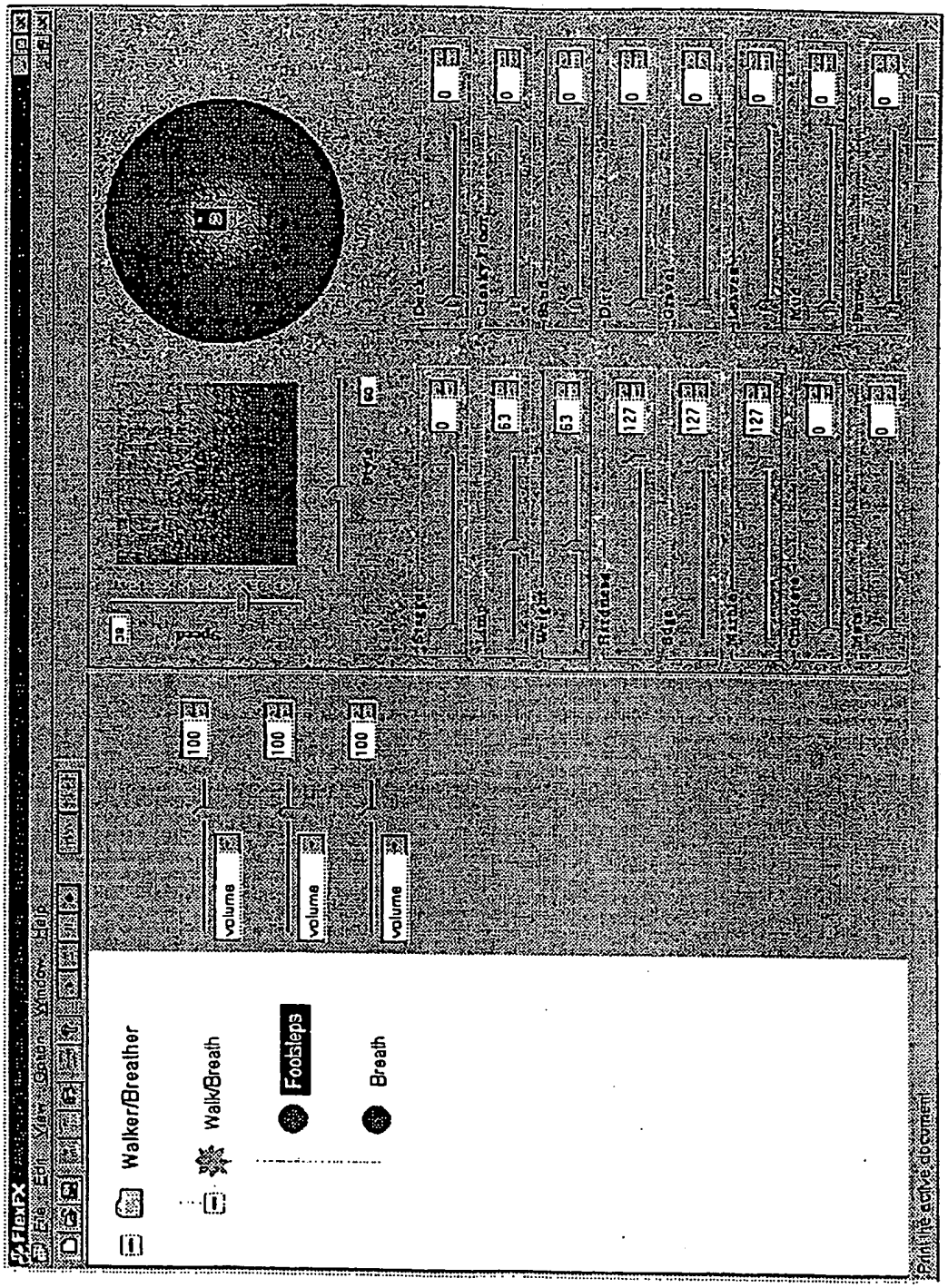

Walker/Breather
Walk/Breath
Footsteps
Breath
volume
volume
volume
100
100
100
Print the active document

FIG. 3

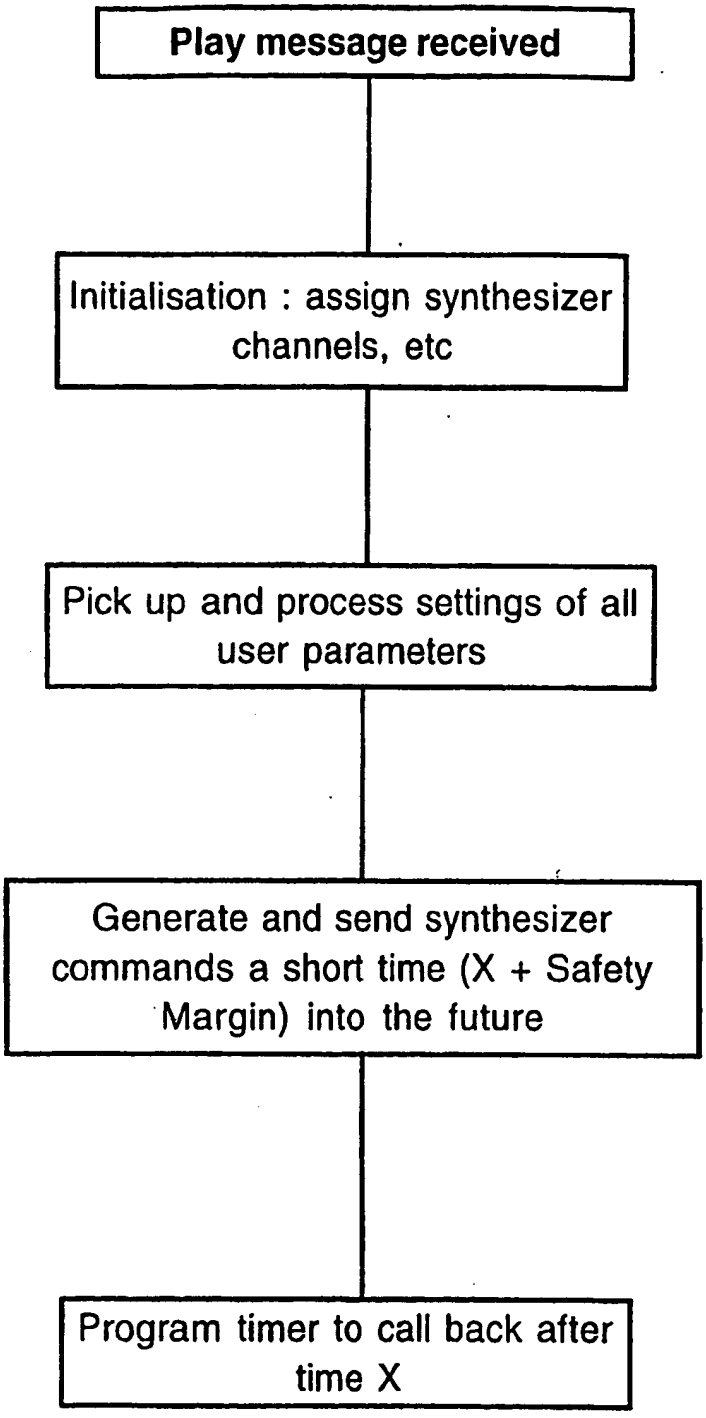

Play message received
Initialisation : assign synthesizer channels, etc
Pick up and process settings of all user parameters
Generate and send synthesizer commands a short time (X + Safety Margin) into the future
Program timer to call back after time X

FIG. 4A

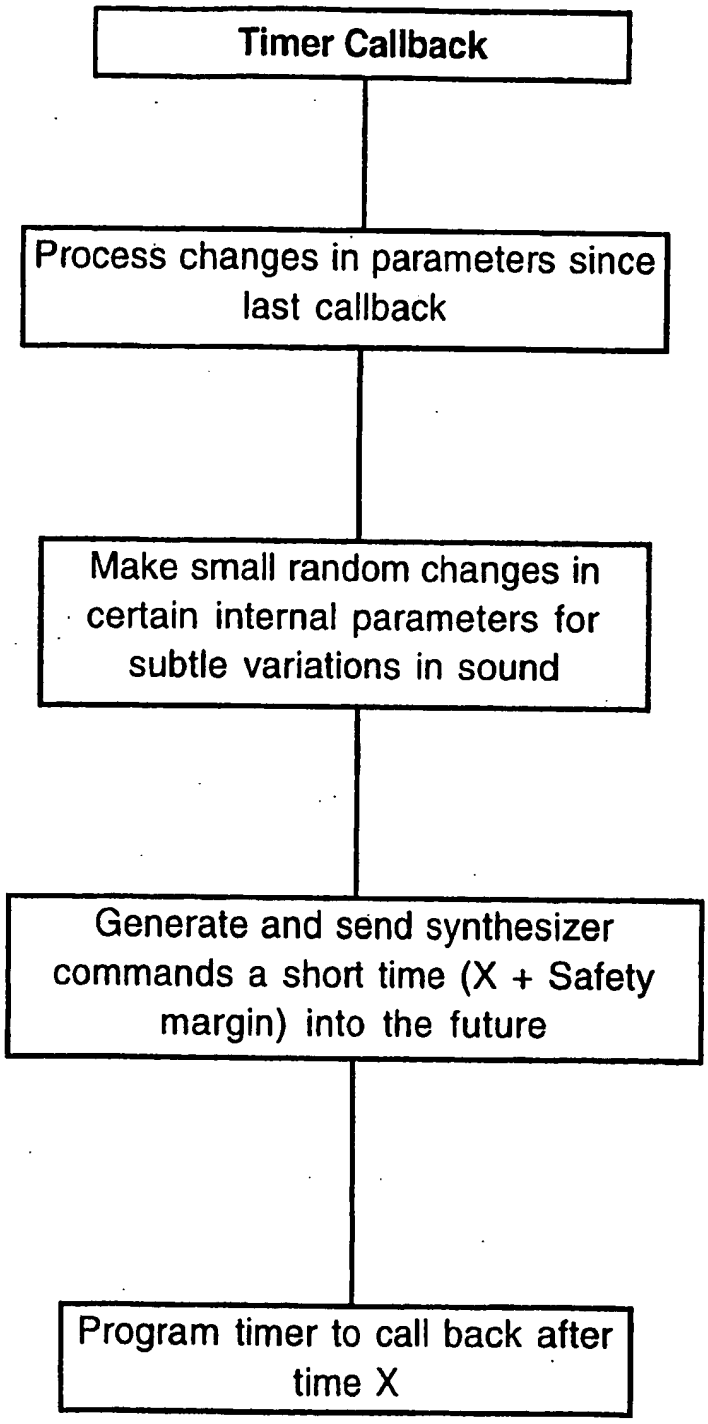

Timer Callback
Process changes in parameters since last callback
Make small random changes in certain internal parameters for subtle variations in sound
Generate and send synthesizer commands a short time (X + Safety margin) into the future
Program timer to call back after time X

FIG. 4B

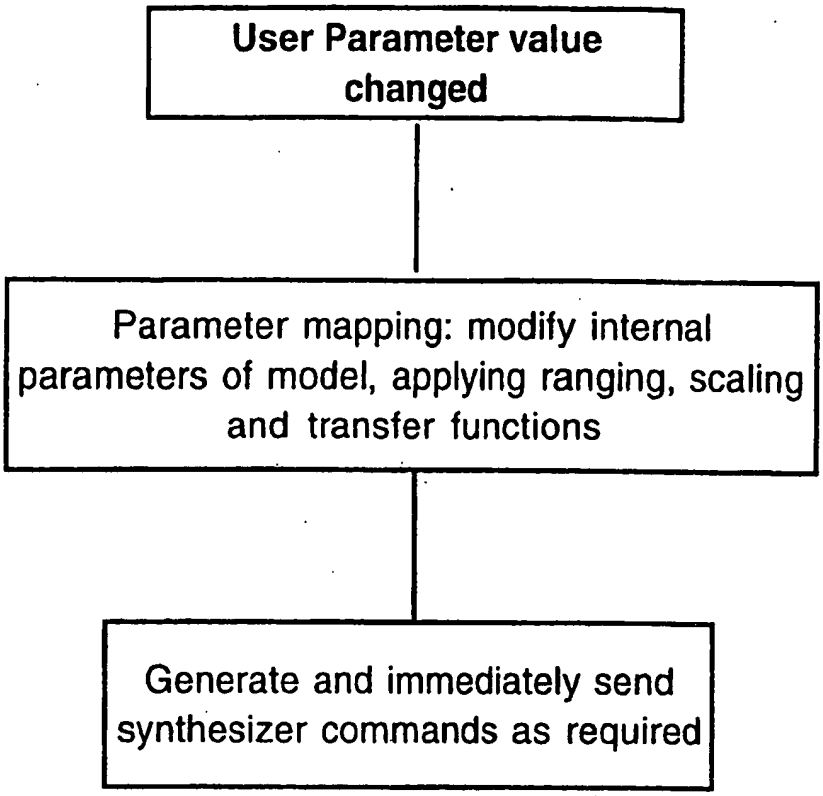

User Parameter value changed
Parameter mapping: modify internal parameters of model, applying ranging, scaling and transfer functions
Generate and immediately send synthesizer commands as required

FIG. 4C

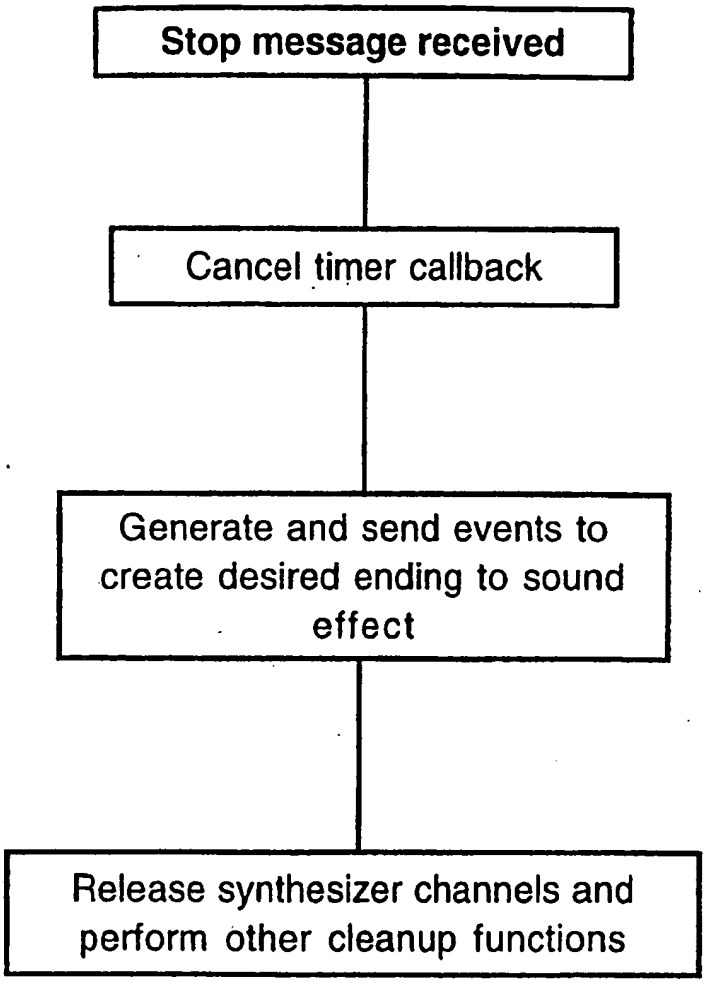

Stop message received
Cancel timer callback
Generate and send events to create desired ending to sound effect
Release synthesizer channels and perform other cleanup functions

FIG. 4D

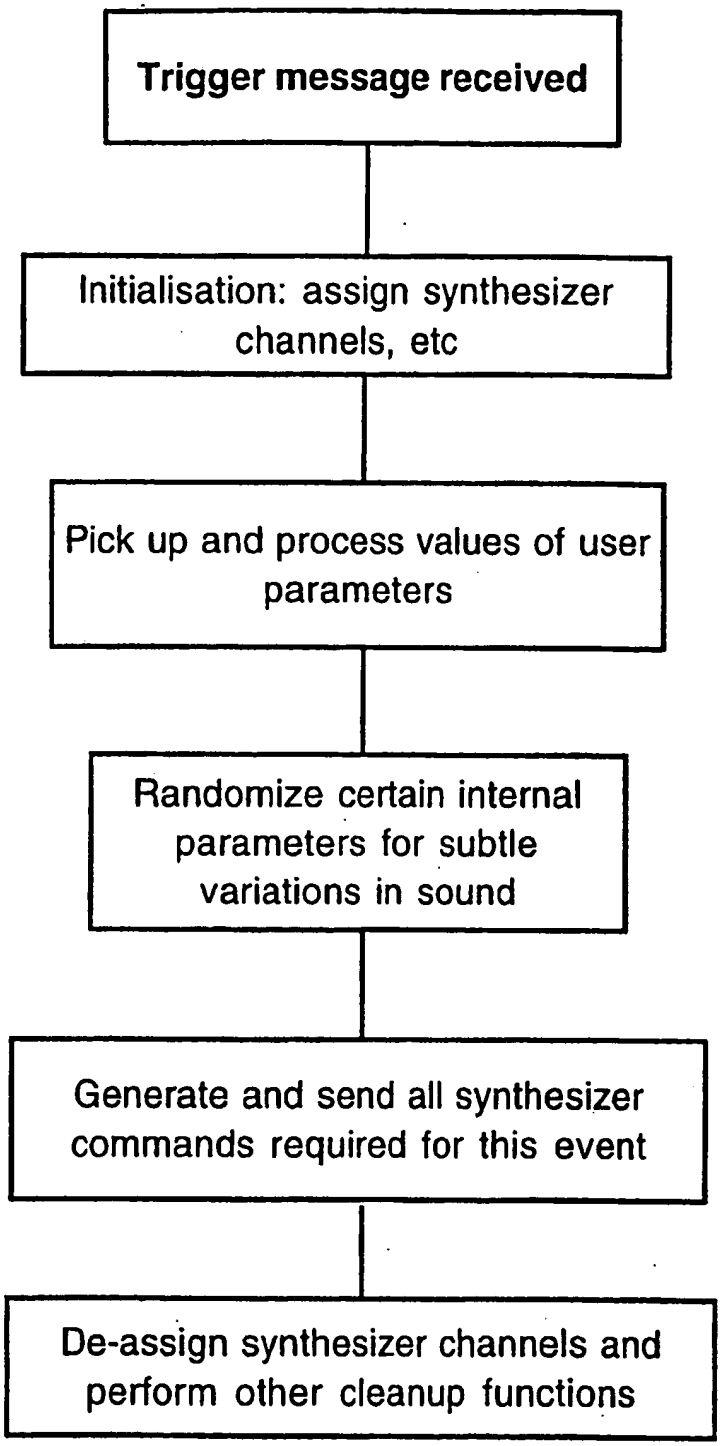

Trigger message received
Initialisation: assign synthesizer channels, etc
Pick up and process values of user parameters
Randomize certain internal parameters for subtle variations in sound
Generate and send all synthesizer commands required for this event
De-assign synthesizer channels and perform other cleanup functions

FIG. 4E

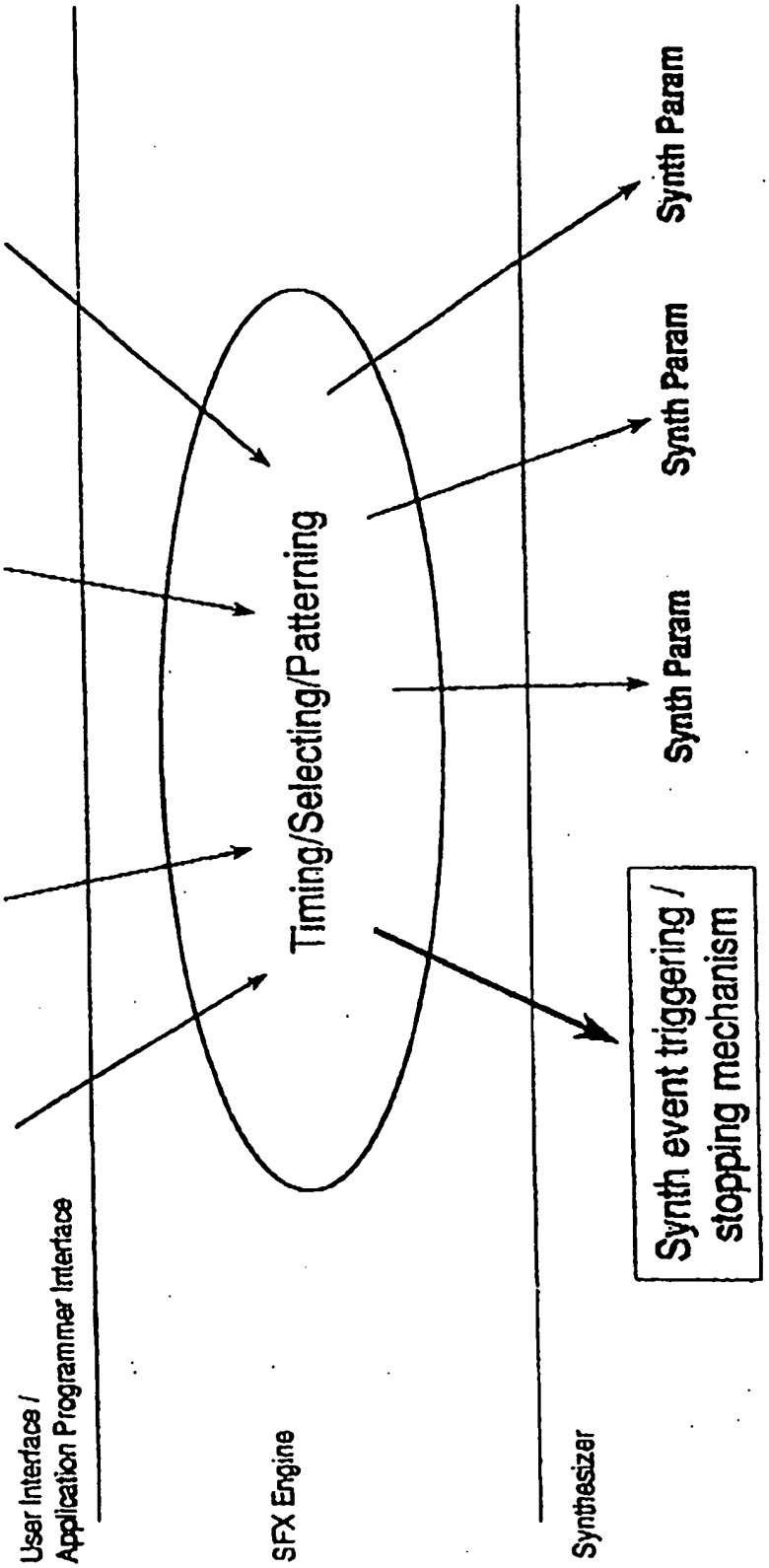

User Param
User Param
User Param
User Param
Timing/Selecting/Patterning
Synth Param
Synth Param
Synth Param
Synth event triggering / stopping mechanism
User Interface / Application Programmer Interface
SFX Engine
Synthesizer

FIG. 5

Footsteps Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations	x	x	x	x		x		x	x
Location				x	x		x		
Walking Speed			x						
Walking Style			x						
Limp			x	x		x			
Stagger			x	x		x			
Weight	x			x		x			
Hardness							x	x	
Edge									x
Surface Type	x			x					

FIG. 6

Guns Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations	x			x		x		x	
Location				x	x		x		
Rate of firing			x						
Irregularity in firing			x						
Hardness							x	x	
Edge									x
Pitch						x			
Gun type	x			x					
Shell cases	x			x					
Ricochet	x			x					

FIG. 7

Driving Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations		x							
Break Pedal	x		x	x		x			x
Clutch Pedal	x		x	x		x			x
Gas Pedal	x		x	x		x			x
Location				x	x		x		x

FIG. 8

Applause Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations	x								
Number of clappers		x							
Clapper Enthusiasm			x	x					
Clapper Hand Size						x	x	x	
Location				x	x		x		x

FIG. 9

Stream Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations		x							
Stream Size	x		x	x		x	x	x	
Location				x	x		x		x

FIG. 10

Machine Drone Models

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations	x	x							
Location				x	x		x		
Brightness						x	x		
Tone			x			x	x	x	
Flutter				x		x		x	

FIG. 11

Canned Laughter Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations	x	x	x						
Location				x	x		x		
Density	x		x						
Density Variation	x		x						
Spontaneity	x		x						
Enthusiasm	x		x	x		x			
Tall-Off			x						
Tone							x	x	
Gender Ratio	x					x			

FIG. 12

Crowd Cheering Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations	x	x	x						
Location				x	x		x		
Density	x		x						
Density Variation	x		x						
Spontaneity	x		x						
Enthusiasm	x		x	x		x			
Tail-Off			x						
Tone							x	x	
Gender Ratio	x					x			

FIG. 13

Wind Sound Model

User Parameters	SFX Engine Parameters			Synthesizer Parameters					
	Event Selection	Event Patterning	Event Timing	Volume	Pan	Pitch	Filter Cutoff	Filter Q	Amplitude Envelope
Random Fluctuations				x		x	x	x	
Location				x	x		x		
Wind Strength	x			x		x	x	x	
Gustiness				x		x	x	x	
Width of Variation				x		x	x	x	

FIG. 14